An Introduction to PatPilr

Raphaël Jauslin

3 janvier 2019

Table des matières

1 Overview

This package is a tool to facilitate the pre-treatment and the treatment of NGS data. The tools is implemented to work on fastq and fasta file. This introduction will, step-by-step, explain how to use the package and which functions you should use in order to obtain your data merged, demultiplexed and cleaned. Moreover it explains some useful function that you could used on your fastq or fastq files.

preTreatment will work correctly if you put your files in a folder that contain two files ".fastq" with R1, R2 in the file's name. It should also contain the information need for the demultiplexing step. See Section 2.2.

Firstly, we need to download and install the package PatPilr. You could find the package in the github repository : https://github.com/RJauslin/PatPilr. Depending on your OS the installation is different. If you are on Linux, you should launch the following commands in `R` or `Rstudio` in order to install PatPilr.

```
install.packages("devtools")
devtools::install_github("Rjauslin/PatPilr@master")
library(PatPilr)
install.PatPil()
install.flash()
```

Linux : It is possible that you have never installed the package **devtools**. If this is the case, it is possible that you have some errors during the installation of the packages. You need to install the dependencies of the packages. Open a terminal and copy paste the following command.

```
sudo apt-get install build-essential libcurl4-gnutls-dev libxml2-dev libssl-dev
```

Windows . You need to install Rtools. Go to the URL https://cran.r-project.org/bin/windows/Rtools/ and choose Rtools35.exe. Follow the setup instructions and when given the option to edit your PATH, take it.

2 Pre-treatment

This is the first step of the pipeline. The inputs are the original files from the server. They are probably named something like `xxx_R1.fastq` and `xxx_R2.fastq`. We will explain how the program deal with the main three steps, merging, demultiplexing, and cleaning. During all the process we will suppose that you have put only the two fastq files and the information needed for the demultiplexing (see Section 2.2).

```
library(PatPilr)

#Linux
pathFolder <- "/home/raphael/Documents/......./working_directory/"
#Windows
pathFolderWindows <- "C:/Users/raphael/......./working_directory/"
```

All the pre-treatment is wrap inside a function that call the different programs. So you only have to check the parameter of the main function `preTreatment`.

```
preTreatment(pathFolder,
  m = 10, # min overlap
  M = 100, # max overlap
  x = 0.25, # max mismatch density
  t = 4, # number of threads
  mismatch = FALSE, # allows 1 mismatch in tag if TRUE
  err = 0.01, # expected error
  slide = 50, # sliding window
  minlength = 60) # minimum length of sequences considered
```

2.1 Merging

The merging step currently implemented is done by the program FLASH [Magoč and Salzberg, 2011]. We have allowed some possible change.

m The minimum required overlap length between two reads to provide a confident overlap.

M Maximum overlap length expected in approximately 90% of read pairs.

x Maximum allowed ratio between the number of mismatched base pairs and the overlap length.

t Set the number of worker threads.

2.2 Demultiplexing

We will here define the requirements for the demultiplexing part.

2.2.1 Simple tag

In case of simple barcode you should only put one additional file called `barcode.txt`. The demultiplexing step is implemented by the program `PatPil` that is hide in the package. The function calls tools `D_simple_tag` that could be used from the shell by the following command. So it is really important that the barcode file have the right format.

```
./PatPil D_simple_tag -f ./merged.fastq -o ./outputFolder/ -b ./barcodes.txt -mismatch
```

The only thing that you should care is that your `barcode.txt` file is of the following form. **Specifically, it is really important to add the .fastq and not another extension file and you should verify that the separator between the tags and the names of the files is a tab.**

```
ACGAGTGCGT 01.fastq
ACGCTCGACA 02.fastq
AGACGCACTC 03.fastq
AGCACTGTAG 04.fastq
ATCAGACACG 05.fastq
ATATCGCGAG 06.fastq
CGTGTCTCTA 07.fastq
CTCGCGTGTC 08.fastq
...
```

If your barcode file follow all these requirements, the demultiplexing part should work proprely.

2.2.2 Double tag

In case of double barcode you should only put three additional files called `forwardtag.txt`, `reversetag.txt` and `primer.txt`. The forwardtag and reversetag contains the barcodes that it supposed to be at the beginning and at the end of your sequences. **You should not include some extensions file such as .fastq or .fq in these two files**.

```
ACACACAC ForwardTag1
ACGACTCT ForwardTag2
ACGCTAGT ForwardTag3
ACTATCAT ForwardTag4
...
```

```
ACACACAC ReverseTag1
ACGACTCT ReverseTag2
ACGCTAGT ReverseTag3
ACTATCAT ReverseTag4
...
```

The `primer.txt` should contain only two information, that forward primer and the reverse primer **in this order**. The primers could have some uncertain nucleotide. The program transform and do all the combination by creating two new files called primerforward and primerreverse. The transformations are done with the following table.

```
CAAAATCATAAAGATATTGGDAC GAAATTTCCDGGDTATMGAATGG
```

```
R = AG
Y = CT
S = GC
W = AT
K = GT
M = AC
B = CGT
D = AGT
H = ACT
V = ACG
```

If you named your file with the good title, with the right format, the demultiplexing part should work properly.

2.3 Quality check

The quality check currently implemented evaluating the expected error in a sliding window and discarding sequences with more than percentage of error in the worst quality window [De Vargas et al., 2015].

```
call.qualCheck(fastqPath,
        outputFasta,
        t = 0.01, # expected error
        s = 50, # sligind window
        m = 60) # minimum sequence length
```

2.4 Remove 'N'

This function simply removes the sequences of a fasta or a fastq files that contains a nucleotide 'N'.

```
fastaPath <- ".../file.fasta"
outputFasta <- ".../fileWithoutN.fasta"
call.RemoveNfasta(fastaPath,outputFasta)
```

If you want to use the tool PatPil directly then you can by the following command.

```
./PatPil RemoveNfasta -f ".../file.fasta" -o ".../fileWithoutN.fasta"
```

If you want to apply it on a fastq file.

```
fastqPath <- ".../file.fastq"
outputFastq <- ".../fileWithoutN.fastq"
call.RemoveNfastq(fastaPath,outputFastq)
```

If you want to use the tool PatPil directly then you can by the following command.

```
./PatPil RemoveN -f ".../file.fastq" -o ".../fileWithoutN.fastq"
```

2.5 rmSmallSeq fasta

This function simply removes the sequences of a fasta or a fastq files that are smaller than a fixed integer.

```
fastaPath <- ".../file.fasta"
outputFasta <- ".../fileWithoutN.fasta"
call.rmSmallSeqfasta(fastaPath,
        outputFasta,
        sizemin = 80) # minimum size
```

If you want to use the tool PatPil directly then you can by the following command.

```
./PatPil rmSmallSeqfasta -f ".../file.fasta" -o ".../fileWithoutN.fasta" -m 80
```

If you want to apply it on a fastq file.

```
fastqPath <- ".../file.fastq"
outputFastq <- ".../fileWithoutN.fastq"
call.rmSmallSeqfastq(fastqPath,
        outputFastq,
        sizemin = 80)
```

If you want to use the tool PatPil directly then you can by the following command.

```
./PatPil rmSmallSeq -f ".../file.fastq" -o ".../fileWithoutN.fastq" -m 80
```

3 Database

The package PatPilr allows you to trim primers on a reference database. In order to get some 100% of match when you assign your sequences, you have to compare with sequences that have the same "configuration". Hence, you possibly need to trim your reference base with some primers. The package allows you to do it with two functions. The first one called trimBasePR2 [Guillou et al., 2013] and the second trimBase. Trim a reference database could also be useful to improve the amount of time needed to perform the dechimering of your sequences [Edgar, 2016].

3.1 PR2

The `trimBasePR2` have some parameters that you need to right understand.

```
trimBasePR2(pathFile,
        primerForward,
        primerReverse,
        trim = 0,
        l_min = 100,
        l_max = 500,
        keepPrimer = TRUE)
```

pathFile	The output path file. You have to put complete path to the output file i.e `...\pr2Cleaned.fasta`
primerForward	The character string representing your forward primer. i.e `'CYAGTA...CTC'`
primerReverse	The character string representing your reverse primer. i.e `'CRAAG...AYG'`
trim	A scalar integer representing the number of nucleotide that you want allow to be trimmed on the primers.
l_min	A scalar integer representing the minimal length of the sequences considered. Primers not taken into account.
l_max	A scalar integer representing the maximal length of the sequences considered. primers not taken into account.
keepPrimer	A boolean value, if you want to keep the primers with the sequences or not.

3.2 Other base

The `trimBase` is mainly the same as `trimBasePR2`. It differs only by the fact that you can pass the reference database that you want instead of let the function charging the PR2 database. It have some parameters that you need to right understand.

```
trimBase(fastaPath,
        outputFasta,
        primerForward,
        primerReverse,
        trim = 0,
        l_min = 100,
        l_max = 500,
        keepPrimer = TRUE)
```

fastaPath	The input path file. You have to put complete path to the output file i.e `...\dataBase.fasta`

outputFasta	The output path file. You have to put complete path to the output file i.e `...\dataBaseCleaned.fasta`
primerForward	The character string representing your forward primer. i.e `'CYAGTA...CTC'`
primerReverse	The character string representing your reverse primer. i.e `'CRAAG...AYG'`
trim	A scalar integer representing the number of nucleotide that you want allow to be trimmed on the primers.
l_min	A scalar integer representing the minimal length of the sequences considered. Primers not taken into account.
l_max	A scalar integer representing the maximal length of the sequences considered. primers not taken into account.
keepPrimer	A boolean value, if you want to keep the primers with the sequences or not.

4 Dereplication

This function dereplicate your fasta files. You are supposed to put only the fasta files inside the working directory. The function will create two folders for temporary work. The first one is called *derep_ech* and the second one called *derep*. At the end of the program, inside of the *derep* folder, you will find two fasta files. **RC.fasta** contains your passed sequences and **RCNotpassed.fasta** the sequences that have failed the selection.

```
Dereplicate <- function(pathFolder, # path to the working directory
  within = 3, # threshold for the number of occurence of the sequence within a file
  between = 2) # threshold for the number of occurence of the sequence between files
```

References

[De Vargas et al., 2015] De Vargas, C., Audic, S., Henry, N., Decelle, J., Mahé, F., Logares, R., Lara, E., Berney, Ć., Le Bescot, N., Probert, I., Carmichael, M., Poulain, J., Romac, S., Colin, S., Aury, J. M., Bittner, L., Chaffron, S., Dunthorn, M., Engelen, S., Flegontova, O., Guidi, L., Horák, A., Jaillon, O., Lima-Mendez, G., Lukeš, J., Malviya, S., Morard, R., Mulot, M., Scalco, E., Siano, R., Vincent, F., Zingone, A., Dimier, C., Picheral, M., Searson, S., Kandels-Lewis, S., Acinas, S. G., Bork, P., Bowler, C., Gorsky, G., Grimsley, N., Hingamp, P., Iudicone, D., Not, F., Ogata, H., Pesant, S., Raes, J., Sieracki, M. E., Speich, S., Stemmann, L., Sunagawa, S., Weissenbach, J., Wincker, P., Karsenti, E., Boss, E., Follows, M., Karp-Boss, L., Krzic, U., Reynaud, E. G., Sardet, C., Sullivan, M. B., and Velayoudon, D. (2015). Eukaryotic plankton diversity in the sunlit ocean. *Science*, 348(6237).

[Edgar, 2016] Edgar, R. (2016). UCHIME2 : improved chimera prediction for amplicon sequencing. *bioRxiv*.

[Guillou et al., 2013] Guillou, L., Bachar, D., Audic, S., Bass, D., Berney, C., Bittner, L., Boutte, C., Burgaud, G., de Vargas, C., Decelle, J., del Campo, J., Dolan, J. R., Dunthorn, M., Edvardsen, B., Holzmann, M., Kooistra, W. H. C. F., Lara, E., Le Bescot, N., Logares, R., Mahé, F., Massana, R., Montresor, M., Morard, R., Not, F., Pawlowski, J., Probert, I., Sauvadet, A.-L., Siano, R., Stoeck, T., Vaulot, D., Zimmermann, P., and Christen, R. (2013). The Protist Ribosomal Reference database (PR2) : a catalog of unicellular eukaryote Small Sub-Unit rRNA sequences with curated taxonomy. *Nucleic Acids Research*, 41(D1) :D597–D604.

[Magoč and Salzberg, 2011] Magoč, T. and Salzberg, S. L. (2011). FLASH : Fast length adjustment of short reads to improve genome assemblies. *Bioinformatics*.